What is claimed is:

1. A method of storing application data in a redundant array of independent memories, comprising:

entering the application data in a first memory;

parking the first memory when the first memory has received the application data;

unparking a second memory;

entering the application data in the second memory; and

parking the second memory when the second memory has received the application data, thereby reducing the likelihood of losing application data due to physical impacts to the memories.

2. The method of claim 1, wherein the first memory is initially parked, and then unparked prior to receiving the application data.

3. The method of claim 1, wherein each of said memories comprises a respective disk drive having a hard disk plate, said respective disk drive capable of being parked by moving an actuator motor control arm on said disk drive to a location removed from a data portion of said hard disk plate, and securing said actuator motor control arm in said removed location to restrain said control arm from contacting said hard disk plate when the memory is subjected to shock.

4. The method of claim 1, wherein said application data is entered into to said second memory by:

entering the application data to an internal controller memory; and

retrieving the application data from said internal controller memory to enter the application memory in the second memory.

5. The method of claim 1, wherein the application data is sent from a processor that comprises a common bus to the first memory along a data path that is removed from said bus.

6. A method of saving application data in a redundant array of independent memories, comprising:

receiving the application data in a controller module;

unparking a first memory;

writing the application data from the controller module to the first memory;

parking the first memory;

unparking a second memory;

writing the application data from the controller module to the second memory; and

parking the second memory, thereby reducing the likelihood of irretrievably losing application data due to impactsdamage to either memory.

7. The method of claim 6, wherein the controller module receives said application data from an application module.

8. A master-slave data management system, comprising:

a processor;

a bus in communication with the processor; and

first and second data paths removed from the bus and providing the processor with communication with first and second memories, respectively;

wherein the processor is programmed to avoid writing application data to the first memory unless the second memory is parked and the first memory is unparked.

9. The system of claim 8, wherein the processor comprises a read/write portion and writes to the first memory through said read/write portion.

10. The system of claim 8, wherein the processor is programmed to avoid writing application data to the second memory unless the first memory is parked and the second memory is unparked.

11. A master-slave data management system, comprising:

a processor; and

first and second memories in communication with the processor;

wherein the processor is programmed to send application data to the first memory, park the first memory when it has received the application data, unpark the second memory, and send the application data to the second memory, and park the second memory when it has received the application data, thereby reducing the likelihood of losing application data due to impacts to a memory.

12. The system of claim 11, further comprising:

a bus in communication with the processor; and

first and second data paths separate from the bus and providing the processor with communication with said first and second memories, respectively.

13. The system of claim 11, wherein the processor includes a read/write portion and writes to the first and second memories through said read/write portion.

14. A method of writing data to a plurality of redundant memories, comprising:

writing the data in succession to each of said memories; and

parking each memory upon completion of each memory's data write and prior to writing the data to the next memory.

15. The method of claim 14, wherein each memory is unparked prior to writing data to each memory.

16. The method of claim 14, wherein parking comprises:

sending a park command from a controller to each memory upon completion of its data write and prior to writing the data to the next memory.

17. The method of claim 16, wherein said park commands are generated internally within said controller.